Dynamic content attacks on digital signatures

Adil Alsaid and Chris J. Mitchell
A.Alsaid@rhul.ac.uk, C.Mitchell@rhul.ac.uk

Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK.

Abstract. Digitally signing a digital document is a straightforward procedure; however, when the digital document contains dynamic content, the digital signature may remain valid but the viewed document may not be the same as the document when viewed by the signer. Other similar problems exist even with 'static' documents, if the appearance of a document can be changed. In this paper, we consider previously proposed solutions for such problems, and propose a new solution. Unresolved issues and problems are also discussed.

Keywords: Digital Signature, e-commerce, e-Document, Security

1 Introduction

Digitally signing a digital document is a straightforward procedure, and digital signatures (see, for example (Menezes et al. 1997, Chapter 11)) are a very important cryptographic primitive. Both national and international standards for signatures exist, including the US Digital Signature Standard (DSS) (National Institute of Standards and Technology 2000), which specifies a suite of recommended algorithms, and two multi-part ISO/IEC standards, ISO/IEC 9796 and ISO/IEC 14888. The main security services that can be provided by a digital signature are: message integrity, origin authentication, and non-repudiation. It is important to note that all the existing standards for signatures, including the DSS and the ISO/IEC standards, are concerned with which algorithms to use and not the form of the data that is signed.

As pointed out by Kain et al. (Kain et al. 2002), digital documents with dynamic content may cause a problem for the digital signature verification process. This paper tries to address some of the problems that arise when

signing digital documents that contain dynamic content. It does not discuss other digital signature security problems such as Trojan Horses or securing the Digital Signature workstation, as discussed, for example, in (Spalka et al. 2001*a*, Weber 1998).

The rest of the paper is organized as follows. Section 2 briefly introduces the problem of signing digital documents with dynamic content. Section 3 discusses possible locations for signature functionality in a computer system. Existing solutions to the problems discussed in Section 2 are introduced in Section 4. A novel solution is discussed in Section 5. Finally, issues and unresolved problems are discussed in Section 6.

2 The Signature Interpretation Problem

In order for a program to generate a digital signature on a data structure, e.g. a document, it must first encode it as a serial string of bits and bytes. It is then expected that the signature will unambiguously commit the signer to the contents of this serialized document. However, ambiguities can arise in the interpretation of the data string when this string can be viewed differently by the signer and the verifier of the signature. That is, it is possible to sign a digital document that changes when viewed at a later time, without invalidating the digital signature. One way in which this problem can arise is when the digital document being signed contains dynamic content.

As an example, suppose that the creator of the digital document is different from the signer. The creator produces the document in such a way that it gives the signer the impression that what he is about to sign is what is being displayed. However, the creator may embed dynamic content, e.g. macros or JavaScript, in the document to change its displayed contents when viewed at a later time. Kain et al. (Kain et al. 2002) describe the problem and gave some examples using MS Word, MS Excel, PDF files, as well as HTML documents.

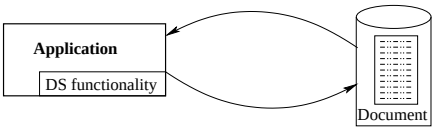
A different source of ambiguities in digitally signed documents was discussed by Jøsang et al. (Jøsang et al. 2002). Jøsang et al. show how font substitution can be used to display the same digital document with different meanings on different computers.

Whilst there are no doubt yet further ways in which ambiguities can be deliberately or accidentally introduced into digital documents, the main focus of this paper is problems arising from dynamic content. This is a significant and growing problem — whether we like it or not, document formats appear to be becoming more complex and more dynamic, rather than less so. Of course, this enables many new features to be provided to users; this appears to be yet another area where user convenience and security are pulling in opposite directions.

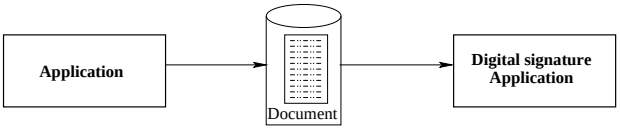
3 Signature functionality

Signature functionality can be integrated into a specific application or implemented as a stand-alone application, see Figure 1. If digital signature functionality is integrated into an application, the application is aware of the document format and could be designed to avoid possible digital signature interpretation issues arising from dynamic content. Moreover, the application could act as a "trusted viewer" for the digital document. However, this is not really a viable general approach, since including signature functionality in every application is potentially very inefficient, with significant associated key management issues.

On the other hand, when a stand-alone signature application is used, the problem of dynamic content can be much more serious, since the digital signature program is typically not aware of the format of the document being signed. One way of avoiding this problem would be to enable the signing application to communicate with the application which understands the document format. This idea forms the basis of the scheme we propose in Section 5 below. Of course, the security of the signing process also relies on the integrity and secrecy of the private signing key and controls to limit its use. The private key must thus be protected in some way, e.g. by storing it in a security module such as a smart card and requiring entry of a password to enable its use.



A. Application integrated Digital Signature functionality



B. Stand alone Digital Signature functionality

Figure 1: Location of digital signature functionality in a computer system

4 Existing Solutions

In this section, previously proposed solutions to the problem of signing digital documents possessing dynamic content are briefly reviewed. Interestingly, all these solutions fall into the second category discussed above, i.e.

they apply to the case where signature functionality is included in a stand-alone application.

4.1 Disabling dynamic content

Disabling dynamic or active content, as proposed by Spalka et al. (Spalka et al. 2001*b*), is one solution to this problem. However, this solution may render some documents useless. Spalka et al. propose two further ways to solve the problem of dynamic content. One is to restrict the actions of active content instead of disabling it, although this would require re-engineering every application. The other approach is to use a 'secure viewer' to view signed documents, but this would require the viewer to be able to parse every possible document format (see also Section 4.4).

4.2 Static file formats

In this approach, only predefined static file formats, known not to have dynamic content, are permitted to be signed. For example, plain ASCII files have no dynamic content, so the digital signature program can sign them without worrying about ambiguity issues. However, this may mean that only one file format can be digitally signed, because most digital document formats permit some sort of dynamic content. This approach may be useful in situations where all digital documents to be signed have no dynamic content features, such as macros, JavaScript, or HTML capabilities.

4.3 XML

Another solution would be to convert the digital document to the Extensible Markup Language (XML) format (W3C 2003) and then apply the XML digital signature processing standard (Eastlake et al. 2002) to obtain the document signature. This does appear to help to solve the problem, but dynamic content may still exist in the XML version. When the document is later presented to the signature verifier, if it is necessary to convert the document back to its original form, the dynamic content may be re-activated.

The authors of the XML digital signature standard are aware of the problem of dynamic or active content. The standard states clearly that, in order to sign an XML document, the signature program should sign all 'external' documents, i.e. documents referenced from within the XML document. The following is a quote from the standard (Eastlake et al. 2002):

> Just as a user should only sign what he or she"sees," persons and automated mechanism that trust the validity of a transformed document on the basis of a valid signature should operate over the data that was transformed (including canonicalization) and

> signed, not the original pre-transformed data. This recommendation applies to transforms specified within the signature as well as those included as part of the document itself. For instance, if an XML document includes an embedded style sheet [XSLT] it is the transformed document that should be represented to the user and signed. To meet this recommendation where a document references an external style sheet, the content of that external resource should also be signed via a signature Reference, otherwise the content of that external content might change which alters the resulting document without invalidating the signature.

One problem with this solution is that the XML document may no longer contain all the dynamic content of the original document. For instance, if a Microsoft Excel document contains macros, then in order to avoid any possible problems arising from such dynamic content, all macros should be removed from the XML version. This will render the document useless if there are macros that are needed to present the document to the user, or if the user wants to make some changes to the document using the macros.

4.4 Document Parser

Another approach to solving the problem is to create a digital signature program with its own document parser. That is, whenever the user wants to sign a document, the digital signature program parses the digital document and removes all dynamic content. In this approach, the digital signature program will need to be aware of most, if not all, digital document formats, which appears infeasible.

Thus, as it stands, this approach is impractical because of the need to provide a document parser for every possible document formats. However, it might be possible to provide a parser for the most popular document format. Nevertheless, problems will still arise since not all document format specifications are available, and the owners of proprietary document formats often change the format with every release of their product.

4.5 Graphics version

The What You See Is What You Sign (WYSIWYS) concept (Scheibelhofer 2001) is designed to solve the ambiguity problem arising from signing digital documents with dynamic content. This approach works by creating a graphical representation of the digital document and then digitally signing it. That is the approach taken by a commercial product (Marketing 2003) running under the Microsoft Windows operating system. It works as follows.

1. When installing the digital signature program, it sets up a special printer driver that functions like a normal printer, but, instead of printing a document on paper, prints it to an image file.

2. The user requests the digital signature program to sign a document by either printing the document to the digital signature special printer from within the application program, or by launching the digital signature program and passing the document as an input. In the latter case, the digital signature program, with the help of the operating system printing subsystem, requests the application program to print the document to the digital signature program special printer.

3. The digital signature program creates a static image of the document, i.e. a graphical representation of the document, using a popular image format, such as TIFF (.tif), bitmap (.bmp), or JPEG (.jpg). It is worth noting that the digital signature program does not need to understand the format of the document to be signed. As stated above, the static image is produced by requesting the application program to print the document to the special digital signature printer driver (using the operating system printing subsystem).

4. The user views the static image of the digital document and approves it for signature.

5. The digital signature program then signs the static image of the document. If necessary, the program can also sign the original document and send it with the static image, but, and according to (Marketing 2003), this should not be used as a legal reference.

This approach appears to work well. However, it removes a lot of the flexibility enjoyed in today's business environment. Also, sending an image potentially consumes a lot more bandwidth than just sending the digital document.

5 A New Solution

In this section, we propose a new method to solve the problem of signing digital documents with dynamic content. The solution works in a similar way to the document parser solution outlined in Section 4.4. The main difference is that our proposed solution passes the document parsing task to the document generator program. This removes the need for the digital signature program to be aware of the document format specifications in order to generate a static version of the document, i.e. a version of the document without dynamic content.

Furthermore, the solution is flexible in that it can handle document formats introduced after the signing program was released. The solution as

described here uses the Microsoft Component Object Model (COM) architecture (Box 1998); however, other component based architectures, such as CORBA or Java, could also be used. The solution is based on two assumptions, as follows.

1. The verifier has access to the program that was used by the signer to generate the digital document. In other words, both signer and verifier have access to the COM object that can generate a 'safe' digital document for the specific digital document type. For example, if the signer is signing a document created by Microsoft Word, then the verifier should also have access to Microsoft Word.

2. All programs that generate digital documents that may need to be signed must be aware of the digital signature program, i.e. they must possess *application awareness*. For example, in the Microsoft Windows environment, this assumption can be met by registering the COM component of the application responsible for creating a static version of the document under a key in the Registry. We will discuss these assumptions in more detail below.

5.1 Application awareness

In order for an application to be digital signature aware, it should meet the following two requirements:

1. It must implement an object that exposes a COM interface to help the digital signature program communicate with the application.

2. When installed, it must register itself in a predefined key location in the Registry, i.e. the data repository in the Microsoft Windows environment in which most of the Windows settings and program information are kept. The Registry location used must be specific to the digital signature program. This will make it easier for the digital signature program to locate digital signature aware applications.

Given that the application meets the above two requirements, the digital signature program can consult the Registry and search for the application that is associated with any digital document (using the file type indication following the full stop in the file name). Once it has identified the application that generated the document, it creates an instance of that application and, using the digital signature COM interface, passes it the document and requests it to generate a static representation of the document. In the next two sections, we describe the processes of signing and verifying digital documents.

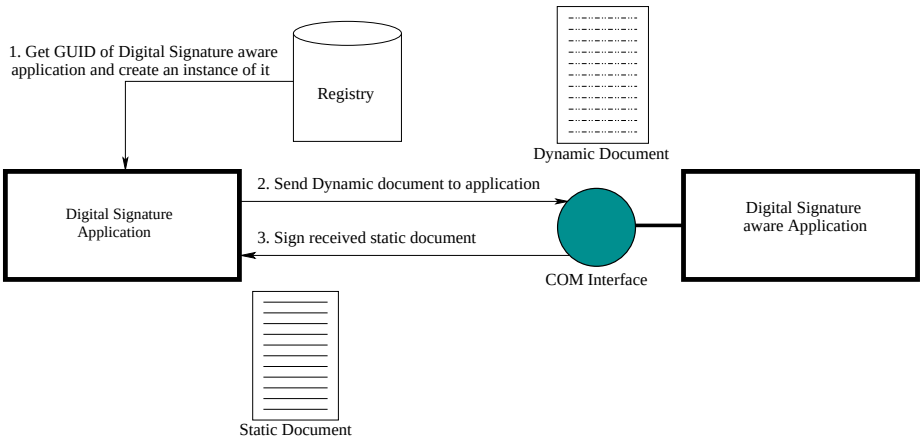


Figure 2: Signing a digital document

5.2 Signing a digital document

To sign a digital document, the signer uses the relevant application to check that the document appears correct. The digital signature program is then invoked and is passed the document. The digital signature program performs the following steps in order to sign the document, as shown in Figure 2.

1. The program consults the Registry and searches for the application program that generated the document, using the document filename extension as a key. It then obtains the Globally Unique ID (GUID) of the application and creates an instance of the application in order to get access to the digital signature interface. If the digital signature program cannot find the GUID of the application responsible for creating the particular document type, the user should be warned, and given the option of either signing the document or not.

2. The program sends the document to the identified application through the digital signature COM interface that was acquired in step 1, and requests it to parse the document and return it in a static form.

3. The signature program receives back the static form of the document and signs it.

5.3 Verifying a signed document

In order to verify a digital signature on a document, the document, the signature, and the signer's public key are input to the signature program for verification. After performing steps 1 and 2 as described in Section 5.2, the signature program verifies the digital signature against the static version of the document it received in step 2 and outputs a 'true/false' indicator. If

the output value is true, then the signature is valid. Figure 3 illustrates the process of verifying a digital signature on a document with dynamic content.

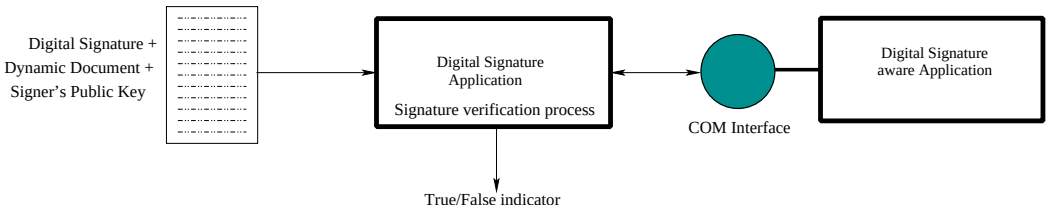


Figure 3: Verifying a signed document

5.4 Security Analysis

We now briefly review some possible attacks on the scheme described immediately above.

5.4.1 File type attacks

As discussed in Section 5.1, the application program must register the file type extensions that it uses in a special location within the Registry, in addition to the 'regular' extension registration process. Correct operation of the proposed solution relies heavily on the correctness of both document extensions and the file type/extension table held in the Registry. Apart from ensuring that the application program possesses application awareness of the digital signature program, the use of a special extension mapping table minimises the risk of accidental changes to this table.

The document extension scheme could be attacked by taking advantage of this reliance. One attack of this type would be to change the extension of a document that is to be signed. For example, suppose that a document is in Microsoft Word format, i.e. it has the extension .doc, and that a malicious third party changes its extension to .txt, the extension for text files. In order to sign the document, the digital signature program performs all the steps discussed in section 5.2, and passes the document to the application registered for handling text files. Since .txt files cannot contain dynamic content, the application will simply return the unchanged file to the signature program, which will sign it.

If an attacker can then change the document type back to .doc before it is viewed by the signature verifier, then problems can clearly arise. If the file contains dynamic content then the problem that the solution was designed to avoid will recur on the verifier's computer. The only way of avoiding this problem is to prevent changes to the file type extension, which can be achieved by including the file name within the scope of the digital signature. However, even if such a precaution is enforced (and this would be

our recommendation) problems can still arise if the extension/application mapping table in the Registry can be modified, as we now describe.

Suppose an attacker can modify the signature program extension/application association tables in the Registry of both the signer's and the verifier's computer so that in both cases .doc files are processed by an application designed to work with ASCII text files. Suppose, moreover, that the signer is given a document to sign that contains dynamic content. When the signature program passes it to the application to make a static version, no changes will be made since the document will be treated as an ASCII text file. Exactly the same will happen at the verifier, and the signature on the document will thus be verified. However, when the verifier views the document using Word, the dynamic content will be activated, and the usual problems with dynamic content arise.

It should be noted that, as long as the file name (and hence the extension) is signed, attacks require the modification of settings on the signer and/or verifier machine. The use of a special association table, used only by the signature program, will prevent such changes being made accidentally. However, no system can completely address threats which arise if attackers have access to the signer or verifier computer, and thus users of signatures should take all the usual precautions to protect the integrity of their computers.

5.4.2 Changes to documents

In order to sign a digital document, the user views the document on the screen, approves it for signature, and finally requests the digital signature program to sign it. However, a threat exists that the document could be changed after the user views it and before the document is signed. For instance, just after viewing the document and before signing it, a piece of malicious code could change the document.

This issue can be addressed by integrating the digital signature functionality into the application itself, instead of separating the viewing and signing functions. An application may provide both facilities to the user; for instance, the application may enable the user to view the document, approve it for signature, and have the signature generated (e.g. using a system function call) without switching to any other application.

Of course, this problem arises with any scheme designed to sign documents, independently of the solution described in this paper. Again, this underlines the importance of protecting the integrity of any computer used to create digital signatures.

6 Concluding remarks

The suggested solution requires all document handling applications to possess application awareness of the digital signature program in order to function properly. Every application must implement a COM interface and register itself in the Registry, in a location specific to the digital signature program, to enable the digital signature program to sign the digital document. We conclude this document by discussing one possible area for possible future research.

In order to sign a digital document, the user private key should be accessible to the digital signature program. Securing the user private key is very important to the operation of the suggested solution and, indeed, to any implementation of digital signatures. Where should this key be stored? The use of trusted computing technology (Balacheff et al. 2003), as incorporated into Microsoft's Next Generation Secure Computing Base (NGSCB) (England et al. 2003), may be useful in this context. Further research in this area is required in order to answer such questions.

7 References

Balacheff, B., Chen, L., Pearson, S., Plaquin, D. & Proudler, G. (2003), *Trusted Computing Platforms: TCPA Technology in Context*, Prentice Hall PTR, Upper Saddle River, New Jersey.

Box, D. (1998), *Essential COM*, Addison-Wesley, Boston, MA.

Eastlake, D., Reagle, J. & Solo, D. (2002), 'RFC 3075: (extensible markup language) xml-signature syntax and processing'.

England, P., Lampson, B., Manferdelli, J., Peinado, M. & Willman, B. (2003), 'A trusted open platform', *IEEE Computer* **36**(7), 55–62.

Jøsang, A., Povey, D. & Ho, A. (2002), What You See is Not Always What You Sign, *in* 'The proceedings of the Australian UNIX User Group', Melbourne.

Kain, K., Smith, S. W. & Asokan, R. (2002), Digital signatures and electronic documents: A cautionary tale, *in* B. Jerman-Blazic & T. Klobucar, eds, 'Advanced Communications and Multimedia Security, IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security, September 26-27, 2002, Portoroz, Slovenia', Vol. 228 of *IFIP Conference Proceedings*, Kluwer Academic, Boston, MA, pp. 293–308.

Marketing, U. S. D. T. S. (2003), 'WYSIWYS What You See Is What You Sign'. http://www.utimaco.de/eng/content_pdf/wysiwys.pdf.

Menezes, A., van Oorschot, P. C. & Vanstone, S. A. (1997), *Handbook of applied cryptography*, CRC Press, Boca Raton, Florida.

National Institute of Standards and Technology (2000), *FIPS PUB 186-2: Digital Signature Standard (DSS)*, National Institute for Standards and Technology, Gaithersburg, MD, USA.
URL: *http://www.itl.nist.gov/fipspubs/fip186-2.pdf*

Scheibelhofer, K. (2001), Signing XML Documents and the Concept of 'What You See Is What You Sign', Master's thesis, Institute for Applied Information Processing and Communications, Graz University of Technology.

Spalka, A., Cremers, A. B. & Langweg, H. (2001*a*), The fairy tale of 'what you see is what you sign' — Trojan horse attacks on software for digital signature, *in* S. Fischer-Hübner, D. Olejar & K. Rannenberg, eds, 'Proceedings of the IFIP WG 9.6/11.7 Working Conference', Security and Control of IT in Society-II (SCITS-II), Bratislava, Slovakia.

Spalka, A., Cremers, A. B. & Langweg, H. (2001*b*), Protecting the creation of digital signatures with trusted computing platform technology against attacks by trojan horse programs, *in* M. Dupuy & P. Paradinas, eds, 'Proceedings of the IFIP SEC 2001', Kluwer Academic, Boston, MA, pp. 403–420.

W3C, T. (2003), 'Extensible markup language (XML)'. http://www.w3.org/XML.

Weber, A. (1998), See what you sign: Secure implementations of digital signatures, *in* S. Trigila, A. P. Mullery, M. Campolargo, H. Vanderstraeten & M. Mampaey, eds, 'Intelligence in Services and Networks: Technology for Ubiquitous Telecom Services, 5th International Conference on Intelligence and Services in Networks, IS&N'98, Antwerp, Belgium, May 25-28, 1998, Proceedings', Vol. 1430 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 509–520.